# 2048

## 项目概述

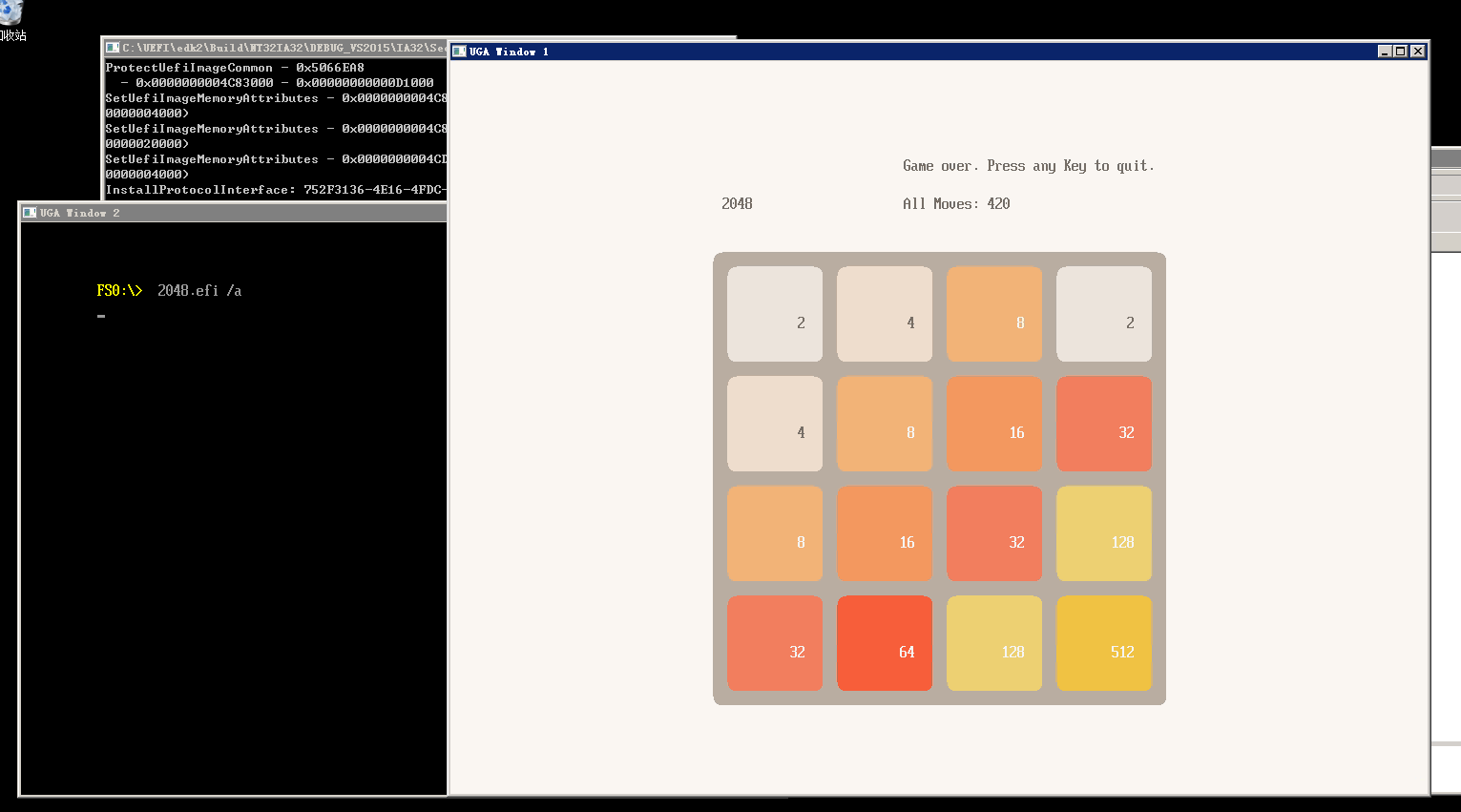
#### 项目简介

这个项目是一个在Uefi环境中可运行的2048小游戏。游戏的最终目标是通过组合， 最终得到2048。而如果最后无法继续移动且未得到2048，则认为玩家失败。

游戏具有图形界面，同时内置了一个简易的AI。玩家可以在启动时选择是由自己游玩还是观察AI。

游戏可以统计步数等信息，也有一些对操作的提醒。

#### 项目截图



## 二、项目结构

#### 1. 项目整体架构

目录：



整个项目包含在一个名为MyPkg的模块中。而主要的源文件则存在子目录2048中，包含编译配置文件2048.inf，棋盘操作部分board.h和board.c，GUI模块gui.h，AI模块ai.c和程序主逻辑部分main.c。

#### 2. 模块定义文件：

###### MyPkg.dec

该文件定义了可以供其他模块使用的数据接口，是必需文件之一。这个文件的内容如下，因为2048使用到了C语言库，因此这个文件的[Includes]部分包含了来自Stdlib的Include目录。



###### MyPkg.dsc

该文件用于编译该Package，也是必需的文件之一。该文件由4部分组成。

Define区定义了该模块相关的全局宏变量：



LibraryClasses区定义了可以引用的库的名字及对应路径，分为入口点类Lib和其他Lib:



这部分我们也并不是特别清楚哪些Lib是必需的而哪些Lib是非必需的，于是就将MdePkg和MdeModulePkg中的大部分Lib都引用了过来。

之后定义要编译的模块：



编译选项：



最后，因为项目使用到了C库函数，还需要再加上一句声明：



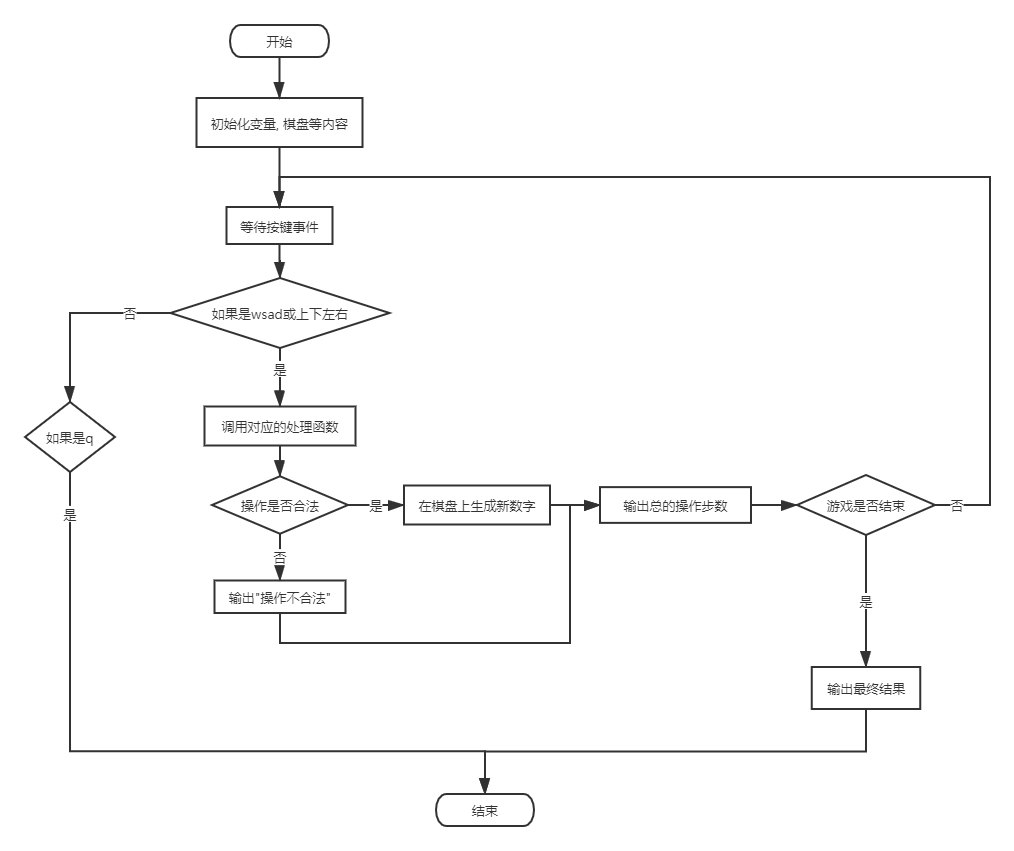
#### 工程声明文件

2048.inf中声明了该工程的基本宏定义，源文件，引用的模块和链接库。因为我们的程序以main为入口点，故声明为Shell应用程序。文件内容如下：



#### 源文件

###### 主要逻辑：main.c



键盘的读取(See P.81, P.105)：



操作判断：



###### 棋盘的实现board.h

棋盘的实现使用了一个unsigned int的二维数组。



下面是board.h中声明的所有函数及其功能



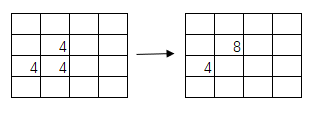
对于初始化棋盘，打印棋盘，计算空位数等均是简单遍历，不再介绍。

判断游戏是否结束，首先先判断当前是否还有空位，如果有的话，即继续。否则，如果有2048，则玩家胜利。如果当前既无空位也无2048，那么判断现在的局面是否可以继续操作。如果无法继续操作了，则玩家失败，否则游戏继续。

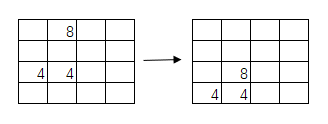
对于具体的棋盘操作，这里以向下举例。

首先调用board\_move\_down，它包括两部分：board\_merge\_down和board\_shift\_down.

这是一个board\_merge\_down的例子：



这是一个board\_shift\_down的例子



组合这两种操作，即可完成所需的操作。

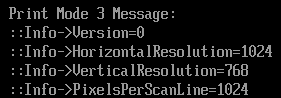
###### GUI的实现

GUI的实现需要自己操作缓冲区。核心api如下：



1. GUI的初始化与销毁

对于一个窗口而言，如果想操作其画面，需要将其指定为（SetMode）图像模式。因此在初始化部分，首先获得两个窗口的句柄（Handle），之后将一个指定为文本模式，一个指定为图像模式。在此过程中也获得了两个文本和图像协议指针作为后续操作的对象。之后，设定图像模式为0x3，此模式下，屏幕分辨率是1024x768。



至于程序结束后的窗口销毁，按照初始化的反向再进行一遍即可。

关键代码如下：



1. 绘图

使用了前面所述的核心api，并将几个高频操作封装成了相应的函数。

1. 单像素



1. 直角矩形



1. 圆角

圆角的实现借助于单个像素点绘图功能。给定顶点坐标以及该顶点的位置以及半径，之后对以半径为边长的正方形的每个像素进行遍历，使用新的颜色重新绘制圆角部分的像素，造成一种圆角被切去的效果。



1. 圆角矩形

圆角矩形的实现是先画一个矩形，之后用背景的颜色画四个角“被切去”的部分。



1. 直接绘制第i行第j列的方块

只是一个简单的封装。



1. 背景的绘制



背景指整个界面不需要发生变化的部分，是其他元素的底板。背景分为三层。最下面是

整个页面的背景，往上一层是棋盘的背景，再向上一层是每个小格的背景。这三层的颜色存储在如下的结构中，并提供了3个宏以便快速取用。



具体的绘制操作也分为三步。先整体的背景，再棋盘背景，之后是16个小块背景。



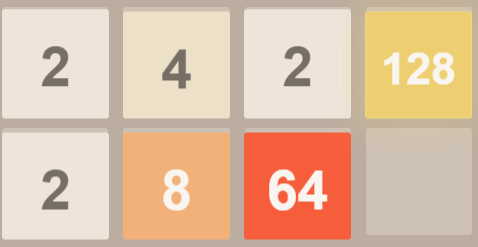
1. 文字显示

文字显示的逻辑实质上是先将文字渲染成位图，之后写入缓冲区。基于HII我们可以封装一个自己的写文字的函数，这里不再详细介绍。



1. “砖块”的渲染

渲染的原则是差异更新，只对发生变化的区块进行重新绘制，以保证应用性能。为此，我们维护了一个棋盘数组的缓存，在渲染时找其中差异化的区块。



另外，不同数值的砖块，其砖块颜色和字体颜色均不同。因此需要使用函数来进行数字->颜色的转换工作。

如下：



1. 其他

除此之外，界面上还有其他的一些部分。例如总步数，无效操作提示等等，这里不再详细阐述。

###### AI的实现

AI实质上只是简单的分数计算，这部分也不是我们所做的重心。

## 三、编译运行



## 四、总结与感想

《UEFI原理与编程》这本书觉得不太行。。除了前两张有相对完整的示例之外，全书其他位置就再也没有完整的示例了。所有需要定义在dsc和inf中的依赖，链接等等一概不提，甚至一些api的头文件是什么都只字不谈。只是简单地罗列了api的原型，但还不如直接看edk源码（实际上我们也确实是这么做的）。此外，还有很多错误，误导和浪费了不少时间。

这个方面的资料太少了，国内几乎找不到相应的材料，就连Google上的英文资料也是少之又少，很多地方还是需要自己去翻源码才写得出来。不过在做项目的过程中，我找到了一个少有的系统性的博客<http://yiiyee.cn/blog/>，还是有不少帮助的。

对于资料少的特点，举个例子如下图，一个常用操作几乎找不到搜索结果。。。

